Load the dataset to work with.

https://sparse.tamu.edu/ is a good starting place.

```
load('filename.mat');
```

Call on the sparse matrix within this dataset.

```
adj_matrix = Problem.A;
```

Transform adjacency matrix to transition matrix, with links between variables acting as edges on a graph. The input argument serves as the adjacency matrix.

```
W = make_transition_matrix(adj_matrix);
```

Construct a PageRank vector with three input arguments for the function: transition matrix W, jump-rate 'alpha' and residual 'epsilon.

```
pr = page_rank(W,0.15,1e-4);
```

```
residual = 1.2406
Iteration #0
residual = 0.3640
Iteration #1
residual = 0.1183
Iteration #2
residual = 0.0386
Iteration #3
residual = 0.0234
Iteration #4
residual = 0.0094
Iteration #5
residual = 0.0061
Iteration #6
residual = 0.0030
Iteration #7
residual = 0.0019
Iteration #8
residual = 0.0011
Iteration #9
residual = 7.1071e-04
Iteration #10
residual = 4.5861e-04
Iteration #11
residual = 3.1905e-04
Iteration #12
residual = 2.2233e-04
Iteration #13
residual = 1.5784e-04
Iteration #14
residual = 1.1276e-04
Iteration #15
residual = 8.1411e-05
Iteration #16
```

Check this sums up to 1.

```
sum(pr)
```

```
ans = 1.0000
```

Construct an Approximate PageRank vector with four input arguments for the function: transition matrix W, starting seed vertex 'seed', jump-rate 'alpha' and residual 'epsilon.

```
approx_pr = approximate_page_rank(W,1,0.15,1e-4);
```

```
ans = 8
Iteration #1
ans = 21
Iteration #2
ans = 45
Iteration #3
ans = 73
Iteration #4
ans = 82
Iteration #5
ans = 93
Iteration #6
ans = 99
Iteration #7
ans = 109
Iteration #8
ans = 114
Iteration #9
ans = 113
Iteration #10
ans = 122
Iteration #11
ans = 115
Iteration #12
ans = 113
Iteration #13
ans = 112
Iteration #14
ans = 108
Iteration #15
ans = 107
Iteration #16
ans = 91
Iteration #17
ans = 88
Iteration #18
ans = 73
Iteration #19
ans = 73
Iteration #20
ans = 67
Iteration #21
ans = 41
Iteration #22
ans = 36
Iteration #23
ans = 27
Iteration #24
ans = 18
Iteration #25
ans = 9
Iteration #26
ans = 8
Iteration #27
```

```
ans = 5
Iteration #28
ans = 2
Iteration #29
ans = 2
Iteration #30
ans = 1
Iteration #31
ans = 2
Iteration #32
ans = 0
Iteration #33
```

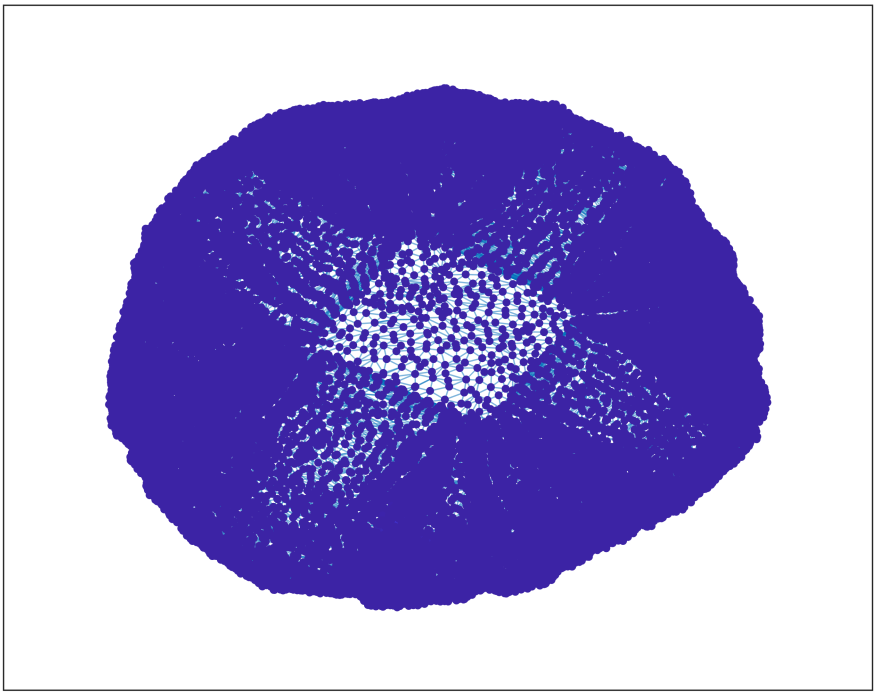
Leave above statement without semicolon if desired to watch function converge.

Now the fun part can begin - colouring of the graph.

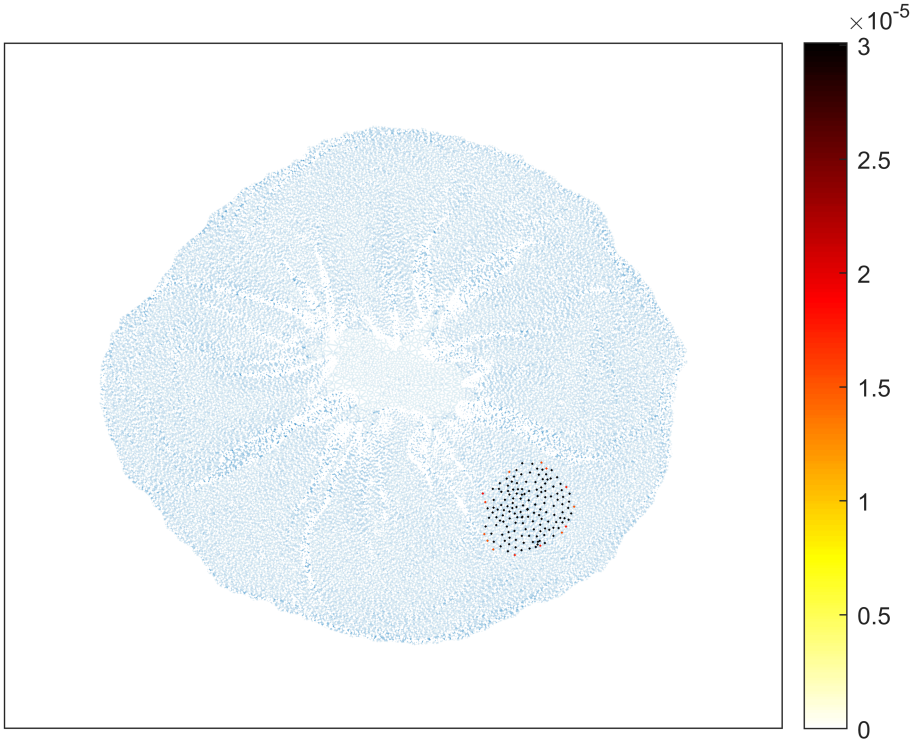
```
G = graph(adj_matrix); %initialise graph
% for directed graph use instead function 'digraph'
figure;
q = plot(G) % plot graph
```

```
q =
  GraphPlot with properties:

     NodeColor: [0 0.4470 0.7410]
    MarkerSize: 2
        Marker: 'o'
     EdgeColor: [0 0.4470 0.7410]
     LineWidth: 0.5000
     LineStyle: '-'
     NodeLabel: {}
     EdgeLabel: {}
         XData: [1×32768 double]
         YData: [1×32768 double]
         ZData: [1×32768 double]

  Show all properties
```

```
bins = approx_pr;
bins = bins + (1+1e-10); % ensure all values 'just' positive
bins = log(bins);
% note that these optional, but must be positive

G.Nodes.value=[approx_pr']; % assign approximate pagerank values
G.Nodes.NodeColors = G.Nodes.value;
q.NodeCData = G.Nodes.NodeColors;
```

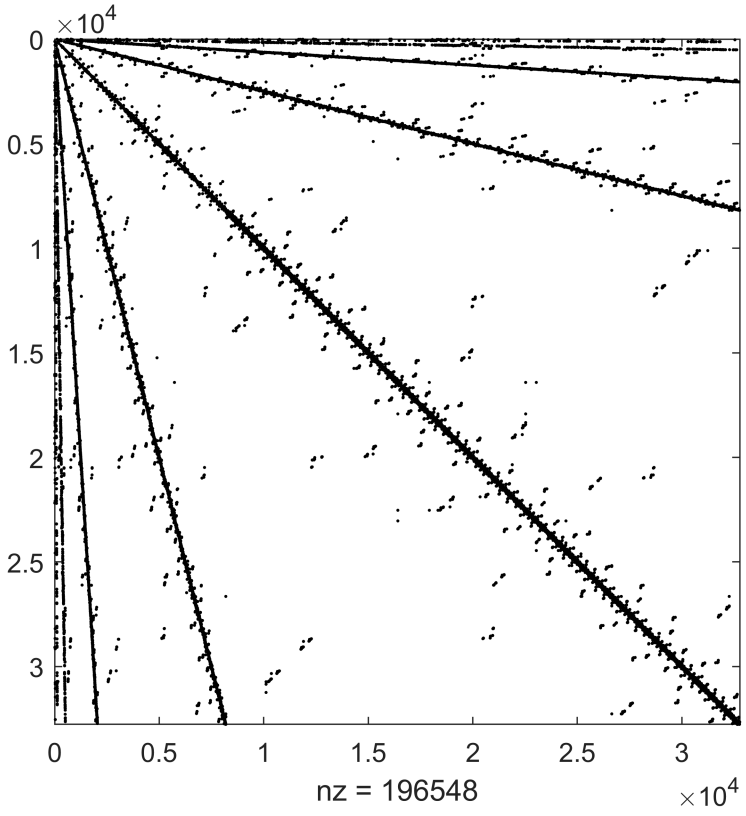


```
figure;
q = plot(G, 'Layout', 'force', 'NodeCData', G.Nodes.NodeColors, 'EdgeAlpha', 0.1, 'MarkerSize',
colorbar; % add colorbar
c = hot;
c = flipud(c);
colormap(c); % add colormap
caxis( [ min(approx_pr) mean(approx_pr) ] ); % optional to format axis
```



Plot the sparsity pattern of adjacency matrix.

```
figure;
spy(adj_matrix);
set(get(gca,'children'),'color','k');
```

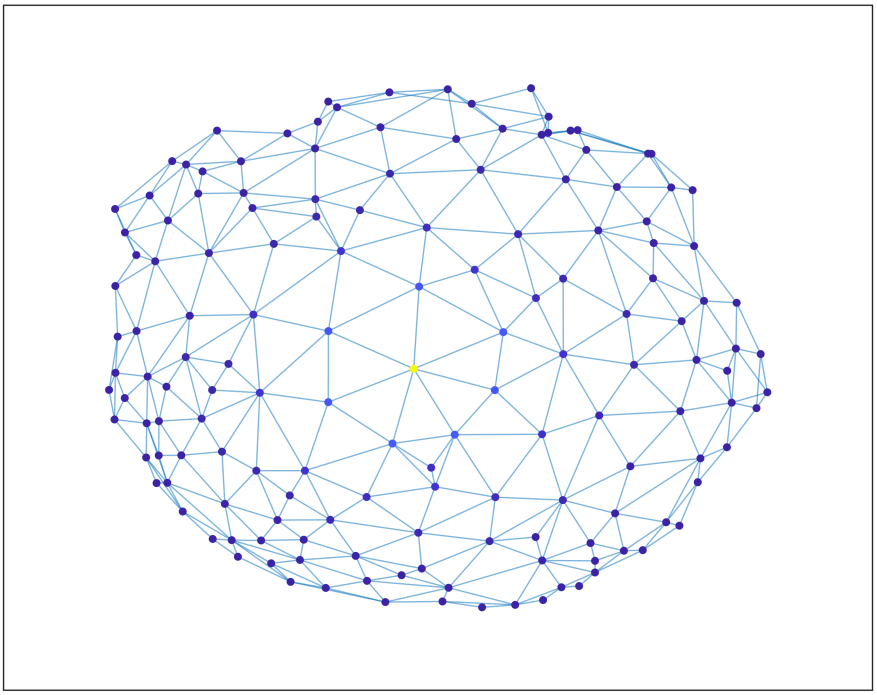


Plot the subgraph for relevant Approximate PageRank matrix.

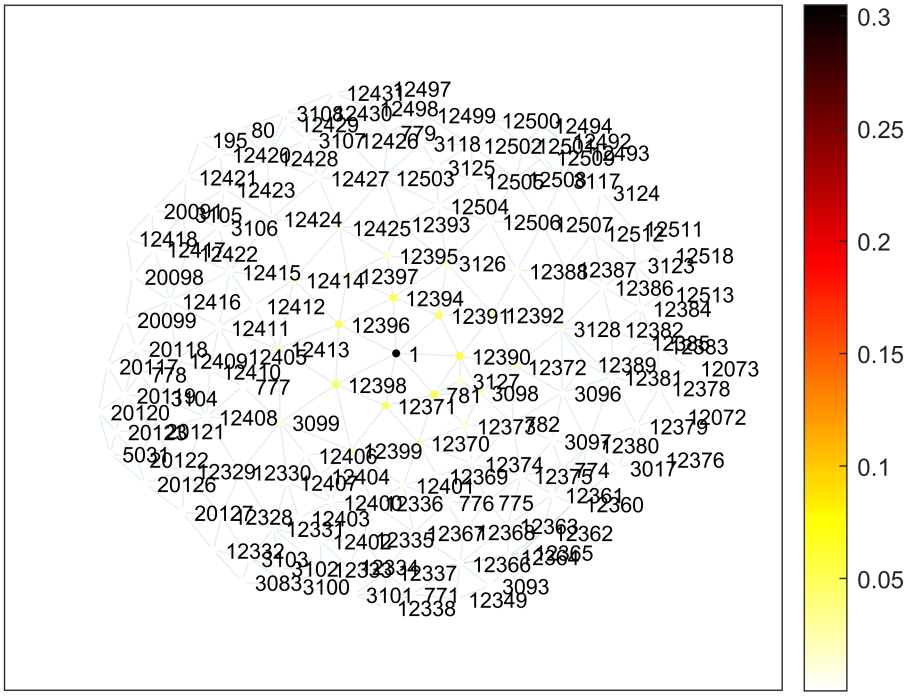
```
bin = approx_pr';
binsize = find(bin);
idx = binsize;
SG = subgraph(G, idx); % assign to nodes the approx.pr values
figure;
sg = plot(SG) % plot subgraph
```

```
sg =
  GraphPlot with properties:

     NodeColor: [0 0.4470 0.7410]
    MarkerSize: 2
        Marker: 'o'
     EdgeColor: [0 0.4470 0.7410]
     LineWidth: 0.5000
     LineStyle: '-'
     NodeLabel: {}
     EdgeLabel: {}
         XData: [1×153 double]
         YData: [1×153 double]
         ZData: [1×153 double]

  Show all properties
```

```
nzero = find(approx_pr);
SG.Nodes.value = nonzeros(approx_pr);
SG.Nodes.NodeColors = SG.Nodes.value;
sg.NodeCData = SG.Nodes.NodeColors;
```



```
figure;
sg = plot(SG, 'Layout', 'force', 'NodeCData', SG.Nodes.NodeColors, 'NodeLabel', nzero , 'EdgeAl
colorbar;
c = hot;
c = flipud(c);
colormap(c);
```



List of function:

```
function[transition] = make_transition_matrix(A)

[m,n] = size(A);

D = A*(1 + 0*[1:m])';

D1 = D;
nonZero = find(D);

D1(nonZero) = (D(nonZero) ).^(-1);

units = 1 + 0*[1:m];
units(nonZero) = 0;

DiagD = diag(sparse(D1));
transition = DiagD*A + diag(sparse(units));


end

%
% pagerank function
%
```

```
function[p] = page_rank(A,alpha,eps)

A1 = make_transition_matrix(A);

[m n] = size(A1);

q = (1/m + 0*[1:m]);

p = q;


% iterations
residual = 10^24;
count = 0; % start from 0 iterations
while (residual > eps)

 p_old =p;
 p = (1-alpha)*p*A1 + alpha*q;

 residual = min( residual, max(abs(p-p_old))*m ) % leave uncommented to watch residual functior

 fprintf('Iteration #%d\n', count);
    count = count + 1; % display # of iterations

end

end

%
% approximate pagerank function
%

function[p] = approximate_page_rank(W,seed,alpha,eps);
oneMinusAlpha = 1-alpha;
[m n] = size(W);

% q = (1/m + 0*[1:m]);

p = sparse(1,m); %page rank

r = p;%residual
r(seed)=1;%make non-zero entity of the one residual node

activeI = seed; %list of active rows
count = 1; % start from 0 iterations
% iterations
while ( ~isempty(activeI) )

  for (i = 1:length(activeI))
   u = activeI(i);

% push_u
   p(u) = p(u) + alpha*r(u); % step(a)
   r_temp  = r(u); % step(b)
```

```
    r(u) = 0.5*oneMinusAlpha*r_temp;

    I_d = find(W(u,:));
    du = length(I_d); %outdegree from node

      for (j = 1:du) % step(c)
          v = I_d(j);
        r(v) = r(v) + 0.5*oneMinusAlpha*r_temp/du;
      end
   end

% update active rows
    i_zero = find(r);
    rPrime(i_zero) = r(i_zero) - eps;
    rPrime = 0.5*(rPrime + abs(rPrime));
    activeI = find(rPrime);
    length(activeI)

    fprintf('Iteration #%d\n', count);
        count = count + 1; % display # of iterations

end


end
```